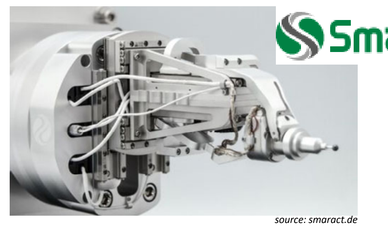
smargopolo: a new control system for SmarGon/MCS2 using ROS

Wayne Glettig, Dominik Buntschu, Ezequiel Panepucci
Swiss Light Source, Paul Scherrer Institut, Villigen, Switzerland

A new control system has been developed for the SmarAct GmbH multi-axis goniometer SmarGon. SmarGon is a six degree-of-freedom positioning device, allowing positioning of a sample and orientation around any given point. It was purpose built for protein-crystallography experiments, but, as will be presented here, was also re-purposed for other applications.
Due to limitations in SmarGon's initial control system, which was based around Delta Tau's PPMAC, a new control system "smargopolo" was developed, based on the open source robotics framework ROS (Robot Operating System) for high-level control, in connection with SmarAct's MCS2 controller for low-level control. The internals of the system will be presented.
This architecture allows strong customisation, mainly regarding interfaces, coordinate systems, logging, debugging and visualisation tools. Also, new calibration routines could be realised, tested and tweaked for optimal practical use.
Because SmarGon's predecessor PRIGo was developed at PSI, several concepts from PRIGo could be reused, allowing tight integration into the overall beamline control software, leading in practice to a better overall reliability of the system.

WHAT IS SMARGON?



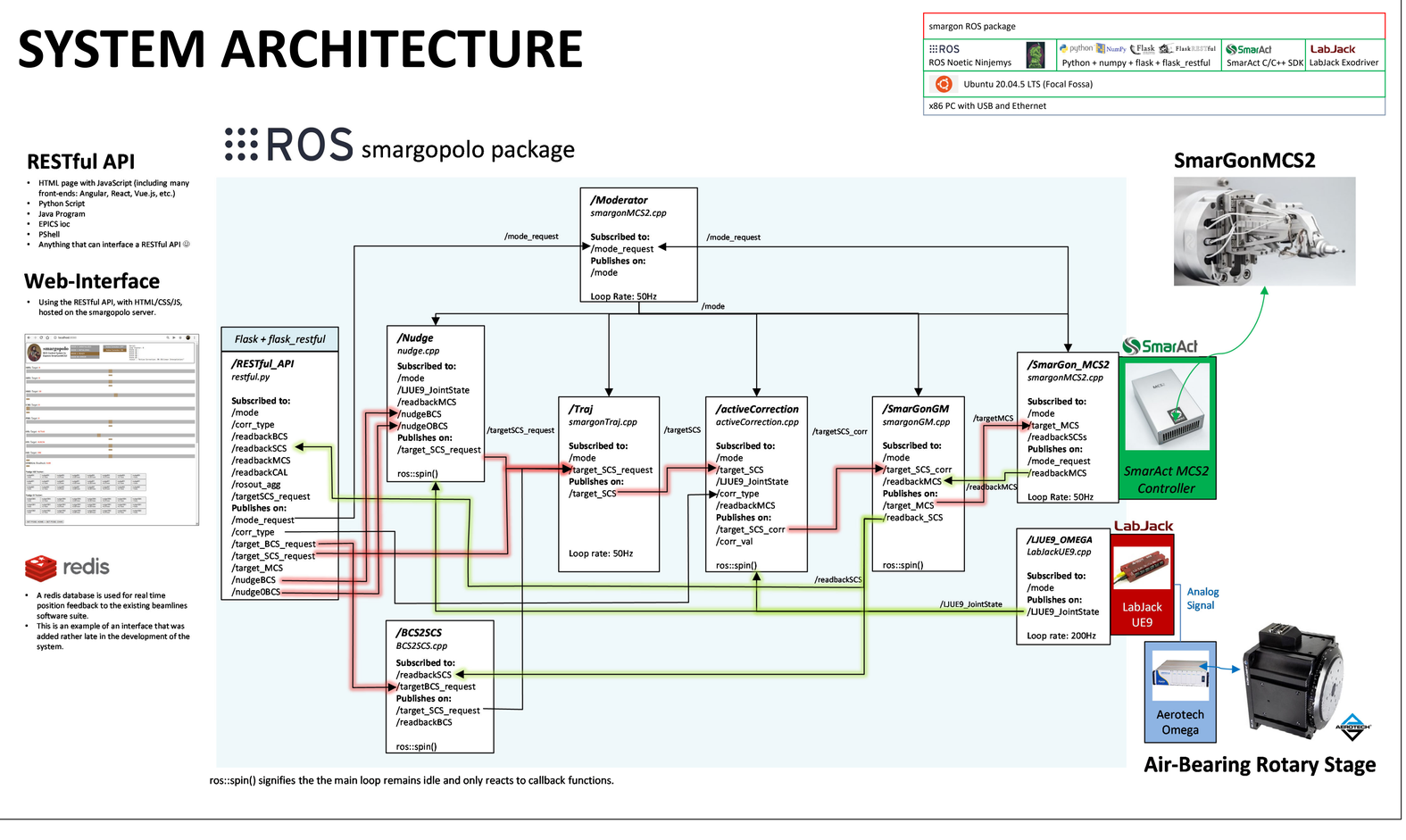
source: smaract.de

Initially developed based upon PSI's 6-axis-goniometer for protein crystallography *PRIGo*, SmarAct GmbH's *SmarGon* is a now commercially available positioning device allowing 4mm translational XYZ motion and three angles of rotation ω: [-∞,+∞], χ: [0,90°], φ: [-∞,+∞] around any arbitrary point in space. Positioning resolution is 1nm, accuracy is below 1µm, and spheres of confusions are achievable of below 3µm.

SmarGon is used to position and orientate a sample w.r.t. the X-ray beam, and is one of the central components of the three SLS protein crystallography beamline setups.

Due to limitations in SmarAct's initial control system, we developed a new control system "smargopolo" using ROS, in connection with SmarAct's newest MCS2 controller for low-level control.

SYSTEM ARCHITECTURE



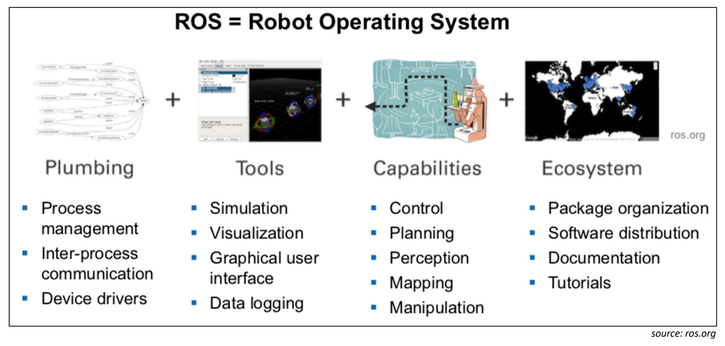
ros::spin() signifies the the main loop remains idle and only reacts to callback functions.

WHAT IS ROS?

ROS Robot Operating System is an open-source software framework, and collection of tools, for robotic software development. Although ROS is not an operating system per se (it runs upon Ubuntu), it provides an environment to develop control systems, leaning upon a microservice architecture, with several inter-process communication models, a build environment, and modular process management.

With packages allowing integration of commonly used robotic concepts like path planning, coordinate transformation, perception, mapping, navigation as well as visualization, simulation, GUI design, and data logging, flexible and powerful systems can be built.

It is widely used in the robotics research community.

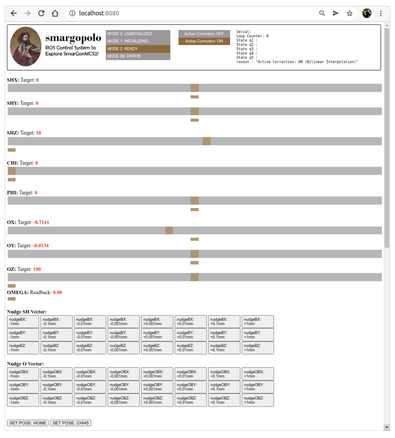


source: ros.org

FLEXIBLE INTERFACES

The use of a PC based control system like ROS allows system interfaces to be precisely tailored to the application environment.
The experimental beamline control software (DA+) communicates with the smargopolo via a RESTful API. Also smargopolo's built-in web-interface (HTML/CSS/JS) communicates over this RESTful API.

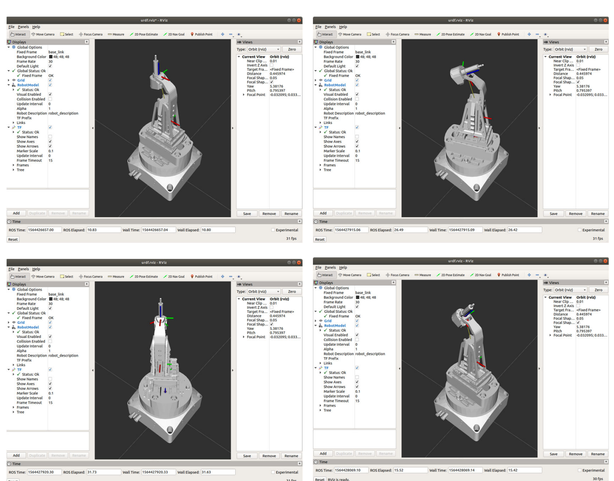
This web-interface allows a fast and direct status overview of the device, access to low level control, without the necessity of beamline overhead.



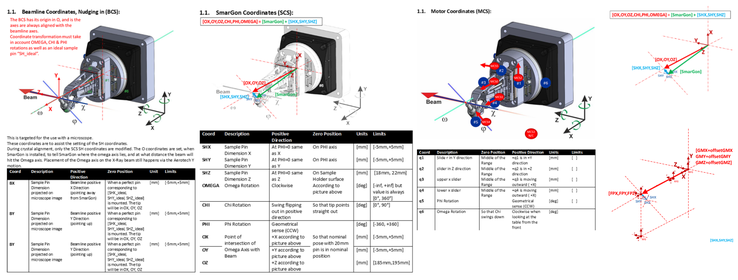
The API can be adapted and extended easily to accommodate new features and use cases, e.g. Tensor Tomography SAXS or Collimator Alignment for spontaneous experiments. For instance, a redis interface was added for fast position readback to an in-memory database.

VISUALISATION TOOLS

ROS comes out of the box with several visualization and simulation tools. A 3D model was used for offline simulation during development or to visualize the current pose of the device when in remote operation.



COORDINATE SYSTEMS

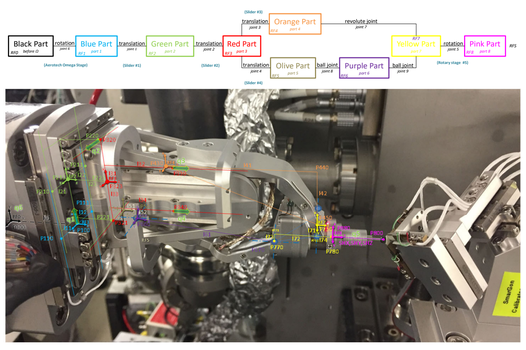


smargopolo uses a geometrical model (smargonGM) to make use of different coordinate systems and the transformations between them. This is very useful to accommodate different use cases.

The following coordinate systems (CS) were implemented:

- Motor Coordinates (MCS)
- SmarGon's CS (SCS)
- Beamline CS (BCS)
- The BCS can be adapted to
  - Microscope CS (imaging view)
  - Calibration Tool CS, if it should be tilted.

Trajectory generation is currently done in SCS.



ERROR HANDLING AND LOGGING

Due to the modular nature of the system, inter-process communication can be monitored, and malfunction or error handling can be controlled from a high-level.

ROS also provides event logging, so coupled with visualization tools, usage patterns can be recorded and replayed or analyzed at a later point in time.

CALIBRATION ROUTINES

To achieve a small sphere of confusion, deviations from the geometrical model to reality were measured, saved and are actively compensated during use. This way, a sphere of confusion of around 1.25µm for ω, and 4µm for χ could be achieved. The performance of the calibration is limited by the mechanical repeatability of the device, which in our setup (temp, vibrations) was around 1µm.

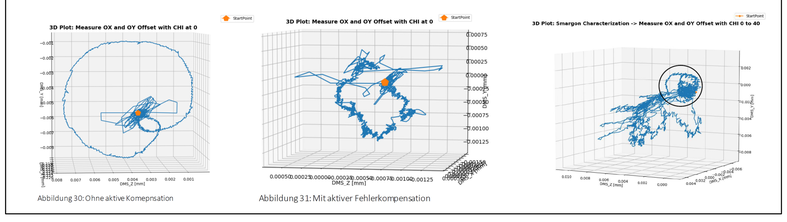


Abbildung 30: Ohne aktive Kompensation
Abbildung 31: Mit aktiver Fehlerkompensation

OTHER USES:

Tensor Tomography SAXS

A special experiment was hosted at the PXI beamline, requiring a tomographic setup and multi axis sample orientation. SmarGon could be re-purposed to allow orientation of the sample along axes commonly used in tomography experiments.

Collimator Alignment

An other special experiment required aligning a 24mm long collimator tube with an inner diameter of 100µm with the beam, to allow the beam to pass through the tube. The entrance of the tube was positioned in the beam, and by setting the rotation centre at this tube entrance of the tube, and orientation could be scanned until the most intense signal could be found.

For NOBUGS Conference 2022, PSI Villigen, CH
Wayne Glettig et al. 18.9.2022

Ultimate goal: Higher reliability and shorter downtimes in case of failure.